AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A method for checkpointing an application, comprising:

dynamically linking an interceptor library into the application at application startup time during a run-time invocation of the application, wherein the run-time invocation occurs after the application has been compiled and linked, and wherein the interceptor library is dynamically linked by simply setting an environment variable, without having to perform an entire static linking process;

intercepting a function call produced by the application at the interceptor library;

recording parameters of the function call to create a checkpoint that includes information about the function call parameters;

making the function call by referring to function pointers saved within the interceptor library;

receiving results of the function call; and

forwarding results of the function call back to the application;

wherein the system records state information without modifying the application or the operating system.

2. (Original) The method of claim 1, further comprising creating a checkpoint by:

stopping the application;

retrieving the recorded parameters;

saving the checkpoint data, including the recorded parameters, to secondary storage; and

resuming the application.

3. (Original) The method of claim 2, further comprising using the checkpoint to restore the application.

4. (Original) The method of claim 2, wherein saving the checkpoint data to secondary storage involves saving the checkpoint data to a persistent storage.

5. (Original) The method of claim 2, wherein saving the checkpoint data to secondary storage involves saving the checkpoint data in a file system, or a database.

6. (Original) The method of claim 1, wherein making the function call involves referencing the function through a function pointer.

7. (Original) The method of claim 1, further comprising recording the results of the function call to facilitate creating a checkpoint that includes information about the results of the function call.

8. (Original) The method of claim 1, wherein the function calls can include system calls or lib calls.

9. (Original) The method of claim 1, wherein the parameters can include:

file paths;

thread flags; and

timer-thread relationships.

10. (Currently amended) A computer-readable storage medium storing instructions that when executed by a computer cause the computer to perform a method for checkpointing an application, the method comprising:

dynamically linking an interceptor library into the application at application startup time during a run-time invocation of the application, wherein the run-time invocation occurs after the application has been compiled and linked, and wherein the interceptor library is dynamically linked by simply setting an environment variable, without having to perform an entire static linking process;

intercepting a function call produced by the application at the interceptor library;

recording parameters of the function call to create a checkpoint that includes information about the function call parameters;

making the function call by referring to function pointers saved within the interceptor library;

receiving results of the function call; and

forwarding results of the function call back to the application;

wherein the system records state information without modifying the application or the operating system.

11. (Original) The computer-readable storage medium of claim 10, further comprising creating a checkpoint by:

stopping the application;

retrieving the recorded parameters;

saving the checkpoint data, including the recorded parameters, to secondary storage; and

resuming the application.

12. (Original) The computer-readable storage medium of claim 11, further comprising using the checkpoint to restore the application.

13. (Original) The computer-readable storage medium of claim 11, wherein saving the checkpoint data to secondary storage involves saving the checkpoint data to a persistent storage.

14. (Previously presented) The computer-readable storage medium of claim 11, wherein saving the checkpoint data to secondary storage involves saving the checkpoint data in a file system, or a database.

15. (Original) The computer-readable storage medium of claim 10, wherein making the function call involves referencing the function through a function pointer.

16. (Original) The computer-readable storage medium of claim 10, wherein the method further comprises recording the results of the function call to facilitate creating a checkpoint that includes information about the results of the function call.

17. (Original) The computer-readable storage medium of claim 10, wherein the function calls can include system calls or lib calls.

18. (Original) The computer-readable storage medium of claim 10, wherein the parameters can include:

file paths;
thread flags; and
timer-thread relationships.

19. (Currently amended) An apparatus that checkpoints an application, comprising:

a dynamic linking mechanism that is configured to dynamically link an interceptor library into the application at application startup time during a run-time invocation of the application, wherein the run-time invocation occurs after the application has been compiled and linked, and wherein the interceptor library is dynamically linked by simply setting an environment variable, without having to perform an entire static linking process;

an intercepting mechanism within the interceptor library that is configured to intercept a function call produced by the application;

a recording mechanism that is configured to record parameters of the function call to facilitate creating a checkpoint that includes information about the function call parameters;

a calling mechanism that is configured to make the function call by referring to function pointers saved within the interceptor library;

a receiving mechanism that is configured to receive results of the function call; and

a forwarding mechanism that is configured to forward results of the function call back to the application;

wherein the system records state information without modifying the application or the operating system.

20. (Original) The apparatus of claim 19, further comprising a checkpoint creation mechanism that is configured to:

stop the application;

retrieve the recorded parameters;

save the checkpoint data, including the recorded parameters, to secondary storage; and to resume the application.

21. (Original) The apparatus of claim 20, further comprising a restoration mechanism that is configured to use the checkpoint data to restore the application to the checkpointed state.

22. (Original) The apparatus of claim 20, wherein the checkpoint creation mechanism is configured to save checkpoint data to a persistent storage.

23. (Original) The apparatus of claim 20, wherein the checkpoint creation mechanism is configured to save the checkpoint data in a file system, or a database.

24. (Original) The apparatus of claim 19, wherein the calling mechanism is configured to make the function call by referencing the function through a function pointer.

25. (Original) The apparatus of claim 19, further comprising a recording mechanism that is configured to record the results of the function call to facilitate creating a checkpoint that includes information about the results of the function call.

26. (Original) The apparatus of claim 19, wherein the function calls can include system calls or lib calls.

27. (Original) The apparatus of claim 19, wherein the parameters can include:

file paths;

thread flags; and

timer-thread relationships.